# Simulink 기반
# 자동차 부품 연계보안 기술 연구

팀명: AutoShield
지도교수: 손준영


팀원: 202029178 홍재왕
202055627 레퐁푸
202055555 석재영

# 목 차

# Ⅰ. 과제 배경 및 목표

## 1. 과제 배경

현대 자동차 산업은 자율주행, 커넥티드카, 전기차 기술의 급격한 발전으로 전례 없는 전자화 시대를 맞이하고 있다. 차량 내부에는 수십 개의 전자제어장치(ECU), 센서, 액추에이터가 탑재되어 있으며, 이들은 CAN(Controller Area Network), LIN, Automotive Ethernet 등의 통신 프로토콜을 통해 실시간으로 데이터를 교환하며 복잡한 제어 기능을 수행한다. 특히, 차량-외부 통신(V2X)과 클라우드 기반 서비스의 확산은 차량 네트워크의 연결성을 더욱 강화하고 있다.

그러나 이러한 고도화된 통신 환경은 새로운 보안 위협을 동반한다. CAN 프로토콜의 낮은 보안성, Ethernet 기반 통신의 외부 노출 가능성 등으로 인해 해킹, 데이터 스푸핑, 악성코드 주입, 리플레이 공격 등의 위험이 증가하고 있다. 실제로, 2015년 Jeep Cherokee 원격 해킹 사건과 같은 보안 사고는 자동차 보안의 취약성을 전 세계적으로 드러냈다. 이러한 위협은 운전자와 승객의 안전뿐만 아니라 자동차 제조사의 신뢰도와 산업 전반에 심각한 영향을 미칠 수 있다.

이에 본 과제는 MathWorks의 Simulink 플랫폼을 활용하여 자동차 전장부품 간 연계 동작을 모델링하고, 부품별 특성에 최적화된 보안 알고리즘(예:AES, ECC, HMAC)을 설계·구현하는 것을 목표로 한다. 다양한 보안 알고리즘의 성능과 적용 가능성을 시뮬레이션 환경에서 비교·검증함으로써, 사용자에게 최적의 보안 대안을 선택할 수 있는 체계적인 프레임워크를 제공하고자 한다. 이는 자동차 보안 기술의 신뢰성을 높이고, 향후 자율주행차 및 커넥티드카의 안전한 상용화에 기여할 것이다.

## 2. 과제 세부 목표

가. Simulink 기반으로 코딩 및 테스트 환경 구축: 자동차 부품의 동작 및 통신을 시뮬레이션할 수 있는 코딩 및 테스트 환경을 구축한다.

나. Simulink 모델링: ECU, 센서, 액추에이터 등 주요 전장 부품을 모델링하여 실제 차량 환경을 반영한 시뮬레이션을 구현한다.

다. CAN 프로토콜 조사: CAN 프로토콜의 메시지 구조, 전송 메커니즘, 보안 취약점을 조사하고, 데이터 무결성과 기밀성을 보장하는 방안을 도출한다.

라. TCP/IP 조사: Automative Ethernet과 TCP/IP 프로토콜의 특성과 보안 요구사항

을 분석하여 V2X 통신에서의 보안성을 강화한다.

마. 보안 알고리즘 구현: AES, ECC, HMAC 등 다양한 보안 알고리즘을 구현하고, Simulink 환경에서 테스트하여 상황별 최적 알고리즘을 선택할 수 있도록 한다.

# Ⅱ. 대상 문제 및 요구조건 분석

## 1. 대상 문제

현대 자동차는 자율주행과 커넥티드카 기술의 발전으로 복잡한 전자제어장치(ECU) 네트워크를 포함한다. 그러나 ECU 간 통신, 특히 CAN 프로토콜 기반의 데이터 교환은 낮은 보안성으로 인해 심각한 위협에 노출되어 있다. 예를 들어, 메시지 위조, 리플레이 공격, 데이터 스푸핑은 차량의 제어 기능을 교란시키며 운전자 안전을 위협한다. 또한, 센서(예: LiDAR, 카메라)와 컨트롤러 단위에서 보안 기능이 마비될 경우, 악성코드 주입이나 데이터 변조로 인해 차량의 오작동이 발생할 수 있다. Automotive Ethernet과 V2X 통신의 확산은 외부 네트워크를 통한 공격 경로를 추가로 생성하며, 이는 차량-클라우드 간 데이터 교환에서의 기밀성과 무결성 문제를 심화시킨다. 이러한 보안 취약점은 차량의 안전성과 신뢰성을 저해하며, 자동차 산업의 지속 가능한 발전을 가로막는 주요 장애물로 작용한다.

## 2. 요구 조건

본 과제는 다음과 같은 요구 조건을 충족해야 한다.

첫째, Simulink 환경에서 차량의 주요 전장부품(ECU, 센서, 액추에이터)과 통신 환경(CAN, Ethernet)을 정확히 모델링하여 실제 차량 동작을 시뮬레이션할 수 있어야 한다.

둘째, 각 부품에 적합한 보안 알고리즘(예: AES-128, ECC, HMAC)을 설계하고 구현하여 데이터 기밀성, 무결성, 인증을 보장해야 한다.

셋째, 다양한 보안 알고리즘을 Simulink 환경에서 테스트하여 성능(연산 부하), 처리 속도(실시간성), 보안성(공격 저항력)을 비교·분석할 수 있는 체계적인 프레임워크를 제공해야 한다.

넷째, 사용자 친화적인 UI(예: MATLAB App Designer 또는 Python GUI)를 개발하여 알고리즘 선택, 적용 결과, 비교 분석 데이터를 시각화(예: 그래프, 표)하고, 사용자가 직관적으로 보안 대안을 평가할 수 있도록 해야 한다.

마지막으로, 부품별 하드웨어 제약(메모리, 연산 능력)을 고려한 경량화된 알고리즘 설계가 필요하며, 추후 시스템 확장성을 보장하기 위해 모듈화된 구현이 요구된다. 이러한 요구 조건은 차량 보안 기술의 실용성과 신뢰성을 확보하는 데 필수적이다.

# Ⅲ. 현실적 제약 사항 분석 결과 및 대책

본 프로젝트는 Simulink를 활용한 자동차 전장부품 보안 기술 연구를 목표로 하지만, 다음과 같은 현실적 제약 사항이 존재한다. 이에 대한 분석과 대책을 제시한다.

## 1. 제약 사항: 제한된 하드웨어 리소스

자동차 전장부품(예: ECU, 센서)은 제한된 메모리와 연산 능력을 가지며, 고성능 보안 알고리즘(예: AES-256, ECC)을 구현하기에 부담이 크다. 이는 실시간 처리 요구사항과 충돌할 수 있다.

**대책:** 경량화된 보안 알고리즘(예: ChaCha, SPECK)을 우선적으로 검토하고, Simulink에서 부품별 리소스 소비를 시뮬레이션하여 최적 알고리즘을 선정한다. 또한, 알고리즘의 모듈화를 통해 필요한 경우 부분적 적용이 가능하도록 설계한다.

## 2. 제약 사항: 복잡한 통신 프로토콜 통합

CAN과 Ethernet(TCP/IP) 프로토콜은 상이한 특성과 보안 요구사항을 가지며, 이들을 Simulink 환경에서 통합적으로 모델링하는 것은 기술적 복잡성을 초래한다. 특히, CAN의 낮은 대역폭은 보안 데이터 오버헤드를 제한한다.

**대책:** Simulink의 Communication Toolbox와 Vehicle Network Toolbox를 활용해 CAN과 Ethernet 통신을 정확히 모델링한다. CAN 프로토콜의 경우, 메시지 인증 코드(MAC)를 추가하여 보안성을 강화하되, 데이터 크기를 최소화한다. Ethernet의 경우, TLS 프로토콜 적용 가능성을 검토한다.

## 3. 제약 사항: Simulink 사용 숙련도 부족

팀원들의 Simulink 사용 경험이 제한적일 경우, 복잡한 부품 모델링과 보안 알고리즘 통합에 시간이 소요될 수 있다.

**대책:** MathWorks의 공식 튜토리얼과 자동차 관련 예제(예: Simulink Control Design)를 활용한 체계적 학습을 진행한다. 초기에는 간단한 ECU 모델부터 구현하고, 점진적으로 복잡도를 높인다. 또한, 팀 내 지식 공유 세션을 통해 학습 효율성을

극대화한다.

### 4. 제약 사항: 보안 알고리즘 성능 평가의 객관성

다양한 보안 알고리즘의 성능(속도, 보안성, 리소스 사용)을 객관적으로 비교하기 위해 표준화된 테스트 시나리오와 평가 기준이 필요하다. 그러나 실제 공격 시나리오를 Simulink에서 재현하는 데 한계가 있다.

**대책:** 일반적인 공격 유형(스푸핑, 리플레이, DoS)을 기반으로 Simulink에서 시뮬레이션 가능한 테스트 케이스를 설계한다. 성능 평가는 연산 시간, 메모리 사용량, 공격 저항력(예: 키 길이, 해시 강도)을 기준으로 정량화하며, MATLAB의 데이터 분석 도구를 활용해 결과를 시각화한다.

### 5. 제약 사항: UI 개발의 시간 및 자원 소요

사용자 친화적인 UI 개발은 추가적인 시간과 전문성을 요구하며, 프로젝트 일정에 영향을 줄 수 있다.

**대책:** MATLAB App Designer를 활용하여 간단한 UI 프로토타입을 우선 개발하고, 필수 기능(알고리즘 선택, 결과 시각화)에 집중한다. 추후 Python(Tkinter, PyQt) 또는 웹 기반 UI로 확장 가능성을 열어둔다.

이러한 대책을 통해 제약 사항을 최소화하고, 프로젝트의 성공 가능성을 높일 수 있다. 지속적인 모니터링과 피드백을 통해 제약 사항을 재평가하며, 필요 시 대책을 조정한다.

## Ⅳ. 사전 기술 학습 및 분석 내용 정리

### 1. Simulink 학습 내용

Simulink는 MathWorks에서 제공하는 모델 기반 설계(Model-Based Design, MBD) 플랫폼으로, 복잡한 시스템의 동작을 시각적으로 설계, 시뮬레이션, 검증할 수 있는 강력한 환경을 제공한다. 본 프로젝트에서는 Simulink를 활용하여 자동차 전장부품(ECU, 센서, 액추에이터)과 통신 프로토콜(CAN, Ethernet)을 모델링하고, 보안 알고리즘을 통합한 가상 시뮬레이션 환경을 구축한다. 이를 통해 부품 간 연계 동작과 보안 시나리오를 테스트하며, 다양한 알고리즘의 성능을 비교·분석한다. 학습 내용은 Simulink 환경 구축과 시뮬레이션 워크플로우 설계로 구분된다.

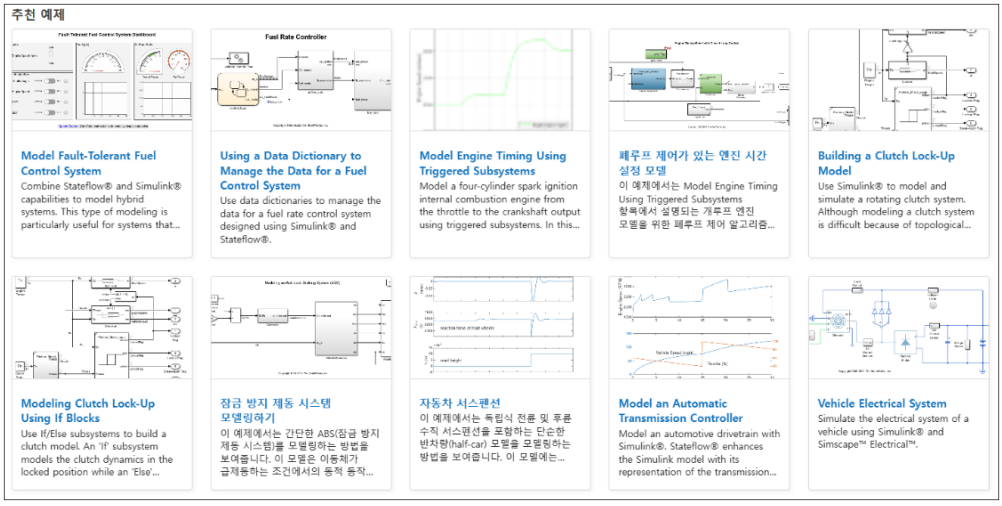


FIG1. Simulink Library

## 가. Simulink 환경 구축

Simulink 환경은 차량 내 전장부품과 통신 시스템을 시뮬레이션하기 위한 기반으로 설정되었다. Simulink Library Browser를 활용하여 Vehicle Network Toolbox, Control System Toolbox 등의 라이브러리에서 블록을 선택해 모델을 구성했다. 전체 시스템은 다음과 같은 흐름으로 설계되었다: 센서 데이터 생성 → 암호화 → CAN/Ethernet 전송 → 복호화 → 출력 시각화. 각 전자제어장치(ECU)는 Subsystem 블록으로 모듈화하여 실제 차량의 부품 구조를 반영했으며, 마스크 아이콘을 설정해 블록 간 시각적 구분을 명확히 했다. 예를 들어, 센서 ECU는 데이터 생성 및 전송을, 제어 ECU는 데이터 수신 및 처리를 담당하도록 구성했다. 또한, Signal Builder와 Constant 블록을 사용해 다양한 입력 시나리오를 생성하며, Simulink Configuration Parameters를 조정해 실시간 시뮬레이션 성능을 최적화했다.

## 나. 시뮬레이션 워크플로우 설계

시뮬레이션 워크플로우는 차량 내 ECU 간 데이터 송수신과 보안 알고리즘 적용 과정을 반영하여 체계적으로 설계되었다. 워크플로우는 다음과 같은 단계를 포함한다:

(1) 암호화 및 복호화 구현: MATLAB Function 블록을 활용해 AES, SPECK 등 경량 보안 알고리즘을 구현했다. 입력 데이터(예: 센서 값)를 암호화하고, 수신 ECU에서 복호화하는 과정을 시뮬레이션했다.

(2) 부품별 단위 테스트: 각 ECU의 Subsystem을 독립적으로 테스트하여 정상 동작을 검증했다. 예를 들어, CAN 통신 블록을 통해 데이터 전송의 무결성을 확인하고, 오류 발생 시 Diagnostic Viewer로 디버깅했다.

(3) 신호 흐름 구조화: Bus Creator와 Bus Selector를 사용해 복잡한 신호 흐름을 구조화했으며, 데이터 송수신 순서를 명확히 정의했다. 이는 CAN의 우선순위 기반 전송과 Ethernet의 고속 통신 특성을 반영한다.

(4) 결과 분석 및 비교: 보안 알고리즘 적용 여부에 따른 성능(지연 시간, 리소스 사용)을 비교하기 위해 To Workspace 블록으로 데이터를 MATLAB 워크스페이스에 저장하고, Scope 및 Display 블록으로 실시간 시각화했다. 또한, MATLAB Script를 연동해 정량적 분석(예: 처리 속도, 메모리 소비)을 수행했다.

다. 프로젝트와의 연계: Simulink 학습은 프로젝트의 핵심 목표인 부품별 보안 최적화와 알고리즘 비교를 위한 기반을 제공한다. Simulink를 통해 CAN/Ethernet 통신 환경을 가상으로 구현하고, 스푸핑, 리플레이 공격 등 보안 위협 시나리오를 시뮬레이션한다. 이를 바탕으로 보안 알고리즘의 실시간 성능과 하드웨어 제약 적합성을 평가하며, MATLAB App Designer로 개발된 UI와 연동해 결과를 시각화한다. 학습 결과를 통해 팀은 Simulink의 모듈화 설계, 통신 시뮬레이션, 데이터 분석 기능을 익혔으며, 이는 후속 단계인 부품 모델링과 알고리즘 통합에 직접 활용된다.

## 2. CAN(Controller Area Network) 통신 프로토콜 학습 내용

가. CAN이란?

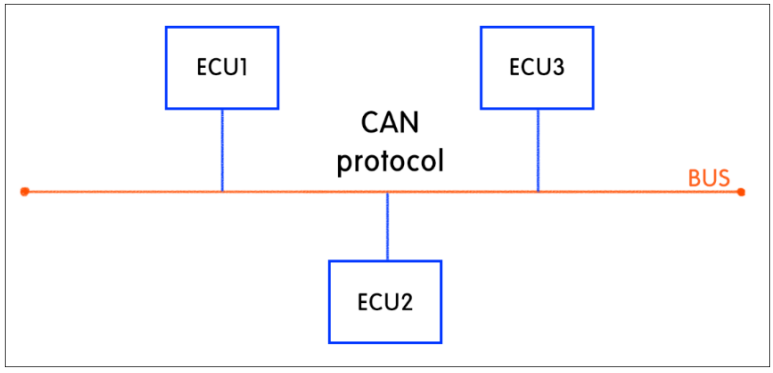


FIG2. CAN Protocol

Controller Area Network(CAN) 프로토콜은 자동차 내 ECU(Electronic Control Unit), 센서, 액추에이터 간 실시간 데이터 교환을 위한 표준 통신 프로토콜로, 본 프로젝트에서 핵심적인 연구 대상이다. CAN은 1980년대 Bosch에 의해 개발되었으며, 낮은 비용, 높은 신뢰성, 실시간성을 바탕으로 자동차 및 산업 제어 시스템에서 널리 사용된다. 본 학습에서는 CAN 프로토콜의 구조, 동작 원리, 보안 취약점, 그리

고 Simulink를 활용한 모델링 방안을 분석했다.

나. CAN 프로토콜의 구조 및 특징

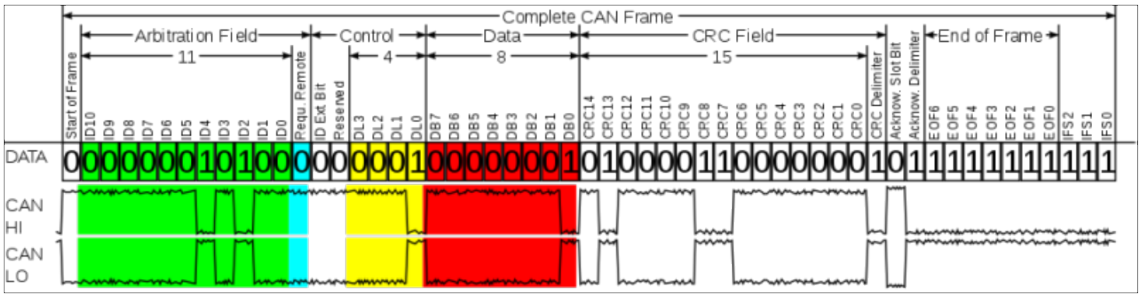


FIG3. 표준 CAN 프레임 포멧

CAN은 멀티마스터, 메시지 중심의 통신 프로토콜로, 최대 1Mbps의 속도를 지원한다. 데이터 프레임은 식별자(Identifier), 데이터 필드(최대 8바이트), CRC(순환중복검사) 등으로 구성되며, CSMA/CR(충돌 해결 방식)을 통해 우선순위 기반으로 메시지를 전송한다. 이는 실시간 제어에 적합하지만, 대역폭 제한으로 인해 보안 데이터를 추가하기 어렵다. CAN은 브로드캐스트 방식으로 동작하므로, 모든 노드가 메시지를 수신하며, 노드 인증이나 암호화 기능이 내장되어 있지 않다.

다. 보안 취약점

CAN 프로토콜은 보안 메커니즘이 부재하여 여러 위협에 노출된다. 첫째, 메시지 위조 공격은 악의적인 노드가 유효한 식별자를 사용하여 거짓 데이터를 전송하는 것으로, 예를 들어 브레이크 제어 신호를 교란할 수 있다. 둘째, 리플레이 공격은 이전에 전송된 메시지를 재전송하여 시스템을 오작동시키는 방식이다. 셋째, DoS(Denial of Service) 공격은 높은 우선순위의 메시지를 지속적으로 전송하여 네트워크를 마비시킬 수 있다. 이러한 취약점은 2015년 Jeep Cherokee 해킹 사건에서 CAN 네트워크를 통한 원격 제어 가능성을 보여주며, 자동차 보안의 시급성을 부각시켰다.

라. 프로젝트와의 연계

본 프로젝트는 CAN 기반 통신 환경에서 보안성을 강화하는 것을 목표로 한다. 이를 위해, CAN 프로토콜의 메시지 구조와 동작을 Simulink의 Vehicle Network Toolbox를 활용해 모델링한다. 예를 들어, ECU 간 데이터 교환을 시뮬레이션하고, 메시지 인증 코드(MAC)나 경량 암호화 알고리즘(예: SPECK)을 적용하여 데이터 무결성과 기밀성을 보장하는 방안을 테스트한다. 또한, CAN의 제한된 대역폭을 고려해 보안 오버헤드를 최소화하는 알고리즘 설계가 필요하다. Simulink를 통해 다양한 공격 시나리오(스푸핑, 리플레이)를 재현하고, 보안 알고리즘의 성능(지연 시간, 리소

스 사용)을 평가한다.

마. 학습 내용 정리

CAN 프로토콜 학습을 통해 메시지 프레임 구조, 우선순위 처리, 오류 검출 메커니즘을 이해했다. 보안 취약점으로는 인증 부족, 암호화 부재, DoS 공격 가능성을 도출했으며, 이를 해결하기 위해 메시지 인증, 데이터 암호화, 이상 탐지 기법을 검토했다. Simulink에서는 CAN 통신 블록을 활용해 ECU 간 통신을 구현하고, 보안 알고리즘 적용 후 네트워크 성능을 분석한다. 이는 부품별 보안 최적화와 다양한 알고리즘 비교를 위한 기초 데이터를 제공하며, 프로젝트의 핵심 목표인 연계 보안 기술 개발에 기여한다.

## 3. TCP/IP 및 Ethernet 통신 기술 학습 내용

가. TCP/IP 통신

TCP/IP 프로토콜은 인터넷 및 로컬 네트워크에서 데이터를 전송하기 위한 핵심 프로토콜 스택으로, 데이터가 네트워크를 통해 전달될 때의 구조와 절차를 정의한다. 일반적으로 4계층 구조(응용, 전송, 인터넷, 네트워크 접근 계층)로 설명되며, 이는 OSI 7계층 모델과 대응된다.

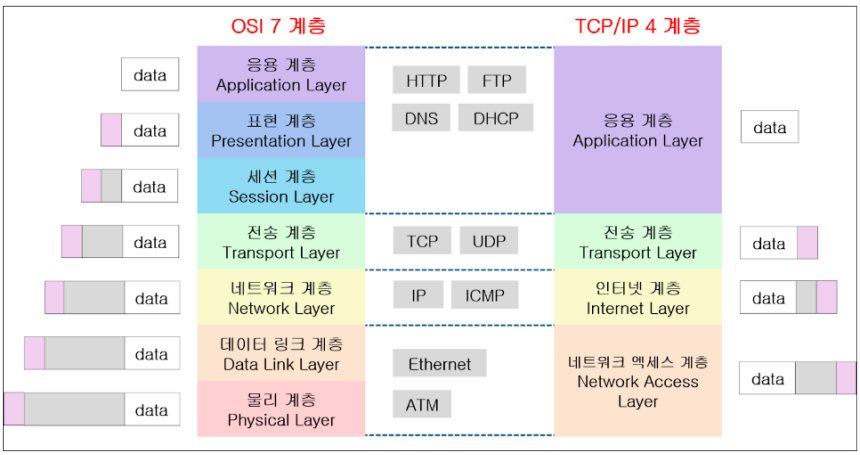


FIG4. OSI 및 TCP/IP 계층 구조

TCP/IP는 크게 두 가지 주요 구성 요소로 나뉜다:

(1) IP(Internet Protocol): 데이터 전송의 기반을 제공하는 비연결형, 비신뢰성 프로토콜로, 데이터를 패킷 단위로 나눠 전송한다. 전송 과정에서 순서 보장이나 오류 복구 기능이 없으며, 이러한 한계는 IP 주소 체계의 한계(특히 IPv4의 주소

고갈 문제)와 함께 차세대 프로토콜인 IPv6 도입의 필요성으로 이어지고 있다.

(2) TCP(Transmission Control Protocol): IP의 한계를 보완하여 신뢰성 있는 데이터 전송을 보장하는 연결형 프로토콜이다. 데이터의 정확한 전송과 순서를 보장하며, 전송 제어 및 오류 복구 기능을 포함하고 있다. TCP는 데이터 연결을 위해 3-Way Handshake 방식(SYN → SYN-ACK → ACK)을 통해 송신자와 수신자 간 연결을 설정한다.

이러한 TCP/IP 통신구조는 차량 내부 네트워크보다는 차량 외부 통신(예:V2X, OTA 업데이트)이나 고속 데이터 전송이 필요한 차량 내부 이더넷 구조에서 활용된다.

나. Ethernet 통신

이더넷(Ethernet)은 근거리(LAN) 및 광역(WAN) 통신망에서 가장 널리 사용되는 유선 통신 기술로, 자동차 분야에서도 고속 데이터 전송이 요구되는 센서 및 ECU 간 통신 수단으로 점차 확대되고 있다. 다음은 네트워크망의 구성도(FIG2)이다.

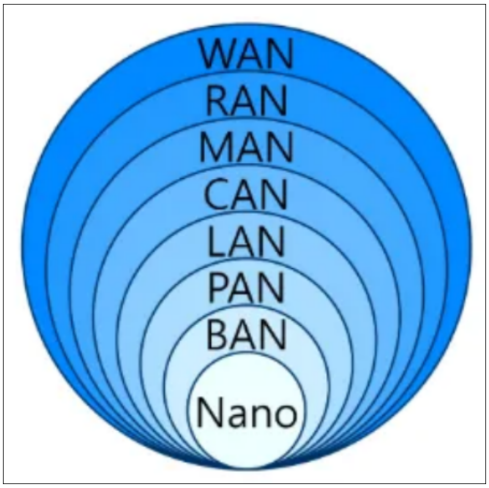


FIG5. 네트워크망의 구성도

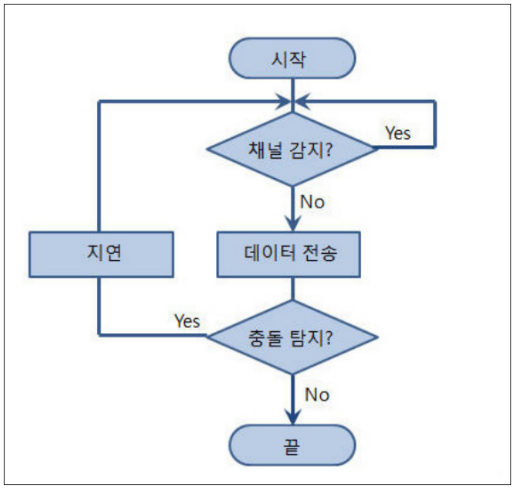


FIG6. CSMA/CD 순서도

이더넷은 OSI 모델의 물리 계층(L1)과 데이터 링크 계층(L2)에 해당하며, MAC(Media Access Control) 주소 기반의 고유 식별 체계를 통해 통신을 수행한다. 이더넷 통신에서는 CSMA/CD(FIG3, Carrier Sense Multiple Access with Collision Detection) 방식이 적용되어 다음과 같은 순서로 데이터 충돌을 감지하고 제어한다.

(1) Carrier Sensing: 전송 회선이 비어(IDLE) 있는지 확인

(2) 전송 중인 데이터가 없으면: 데이터 전송 시작

(3) 전송 도중 충돌 감지 시: 충돌 신호(Collision Signal) 발생

(4) Jam Signal 전송 후: 랜덤한 시간 대기 후 재시도

이더넷은 기본적으로 ACK 응답을 사용하지 않으며, 충돌 발생 시 자체적으로 충돌 감지 후 재전송을 수행하는 구조이다.

차량용 Ethernet 통신의 경우 one-pair UTP(Unshield Twisted Pair) 통신선을 사용하는 구조로 새롭게 설계되었다. 차량 내에 제어 데이터와 멀티미디어 데이터 처리를 위한 모든 요구 사항을 만족하고, 다양한 종류의 데이터를 실시간으로 전달하고 동기화하기 위해 별도의 통신 프로토콜을 개발하였다.

TCP/IP 및 이더넷 통신에 대한 학습은 자동차 보안 알고리즘 설계 시 각 프로토콜의 동작 원리, 한계, 그리고 보안 취약점을 고려하는 데 있어 필수적인 기반이 된다.

# Ⅴ. 개발 환경 및 사용 기술

## 1. 개발 환경

가. 개발 도구

(1) MATLAB/Simulink: 자동차 전장부품(ECU, 센서 등) 모델링과 CAN/Ethernet 통신 시뮬레이션을 위한 핵심 도구. Vehicle Network Toolbox와 Simulink Control Design을 활용해 보안 알고리즘 테스트 환경을 구축한다.

(2) VS Code: Python 및 MATLAB 스크립트 작성과 디버깅을 지원하며, Simulink 모델과 연계된 코드 관리에 사용.

(3) PyCharm: Python 기반 보안 알고리즘 구현과 UI 개발을 위한 통합 개발 환경.

나. 사용 언어

(1) MATLAB: Simulink 모델링, 보안 알고리즘 시뮬레이션, 데이터 분석 및 시각화에 사용.

(2) Python 3.10: 보안 알고리즘 구현, UI 개발, 로그 분석 LLM 파인튜닝에 활용.

(3) Java 17: 서버 측 보안 알고리즘 테스트 및 Ethernet 통신 시뮬레이션에 사용.

## 2. 사용 기술

가. 서버 프레임워크

(1) FastAPI: 비동기 Python 웹 프레임워크로, 보안 알고리즘 테스트 결과를 REST API로 제공하고 UI와 연동.

(2) Spring Boot: Java 기반 서버로, Ethernet 통신 시뮬레이션과 보안 데이터 처리에 사용.

나. 데이터베이스

(1) MySQL: 보안 알고리즘 성능 데이터(연산 시간, 리소스 사용량)와 시뮬레이션 결과를 저장하는 관계형 데이터베이스.

(2) Milvus: CAN/Ethernet 로그 데이터의 벡터화된 패턴을 저장하고, LLM 기반 이상 탐지에 활용.

다. 캐시

(1) Redis: 실시간 로그 데이터 캐싱과 빠른 쿼리 처리를 위해 사용, 특히 LLM 기반 이상 탐지 속도를 향상.

라. AI

(1) PyTorch: 보안 로그 분석을 위한 LLM 파인튜닝과 경량화된 보안 알고리즘(예: SPECK) 구현.

(2) Transformers: HuggingFace의 Transformers 라이브러리를 통해 LLM 모델을 로깅 분석에 활용.

(3) LangChain: 로그 데이터 처리와 LLM의 대화형 분석 워크플로우를 설계.

마. 클라우드

(1) AWS EC2: Simulink 모델과 LLM 실행을 위한 고성능 컴퓨팅 환경 제공.

(2) AWS S3: 시뮬레이션 데이터와 로그 파일 저장.

(3) AWS CloudWatch: 시스템 및 LLM 성능 모니터링.

(4) AWS Lambda: 이벤트 기반 로그 분석 트리거 실행.

바. 배포

(1) Docker: Simulink, Python, Java 환경을 컨테이너화하여 일관된 개발 및 테스트 환경 구축.

사. 로그 수집 도구

(1) Logstash: CAN/Ethernet 통신 로그와 보안 알고리즘 테스트 로그를 수집하고 전처리하여 Milvus 및 Redis로 전달.

아. 화면(UI)

(1) React: 사용자 친화적인 웹 UI로, 보안 알고리즘 비교 결과(성능, 처리 속도, 보안성)를 시각화.

(2) MATLAB App Designer: Simulink 내에서 간단한 알고리즘 선택 및 시뮬레이션 결과 표시 UI 개발.

### 3. 사용 LLM 모델

로깅 시스템에서 고성능 LLM을 활용하는 것은 CAN 및 Ethernet 통신의 이상 패턴 탐지와 보안 위협 분석에 필수적이다. 실시간 로그 분석은 비정상적인 데이터 전송(예: 스푸핑, 리플레이 공격)을 신속히 탐지하고 대응해야 하므로, 효율적이고 정확한 LLM 모델이 요구된다. 본 프로젝트에서는 우선 LLaMA 3 8B 모델을 선택한다. 이 모델은 HuggingFace의 Transformers 라이브러리를 통해 로컬 환경에서 파인튜닝이 용이하며, CAN/Ethernet 로그 데이터의 패턴 분석에 적합하다. 파인튜닝은 PyTorch와 AWS EC2를 활용해 수행하며, 최종 모델은 Docker 컨테이너로 AWS 클라우드에 배포된다. 만약 LLaMA 3의 성능이 로그 분석의 실시간성 또는 정확도 요구사항을 충족하지 못할 경우, HuggingFace에서 제공하는 다른 파인튜닝 가능한 모델(예: BERT, RoBERTa)을 검토하고, Milvus를 활용한 벡터 검색 성능을 최적화한다. 이를 통해 보안 로그 분석의 효율성과 신뢰성을 극대화한다.

## VI. 개발 일정 및 역할 분담

### 1. 개발 일정

개발 일정은 프로젝트의 효율성과 마일스톤 준수를 위해 월별 및 주차별로 구성했다. 주요 마일스톤(착수보고서, 중간보고서, 최종보고서, 발표심사, 결과물 업로드)을 중심으로, 사전 기술 학습, Simulink 모델링, 보안 알고리즘 구현, UI 개발, 테스트

및 보고서 작성을 체계적으로 배치했다. 아래 표는 2025년 5월부터 10월까지의 일정을 나타낸다.

| 기간 | 주요 작업 | 세부 작업 | 담당자 |
|---|---|---|---|
| **5월 1-2주** (5.1-5.16) | 착수보고서 작성 및 제출 | - 프로젝트 개요 및 목표 정리<br>- CAN, TCP/IP, Simulink 사전 학습 계획 수립<br>- 착수보고서 작성 및 지도확인서 제출 | 홍재왕, 석재영, 레퐁푸 |
| **5월 3-4주** (5.17-5.31) | 사전 기술 학습 | - CAN 프로토콜 학습 (프레임 구조, 보안 취약점)<br>- TCP/IP 및 Ethernet 학습 (V2X, 보안)<br>- Simulink 기초 학습 (모델링, 라이브러리 활용) | 홍재왕 (CAN)<br>석재영 (TCP/IP)<br>레퐁푸 (Simulink) |
| **6월 1-4주** (6.1-6.30) | Simulink 환경 구축 및 초기 모델링 | - Simulink 개발 환경 설정 (Vehicle Network Toolbox)<br>- ECU, 센서 모델링 시작<br>- CAN 통신 시뮬레이션 구현<br>- Ethernet 통신 초기 시뮬레이션 | 레퐁푸 (환경 설정, 모델링)<br>홍재왕 (CAN 시뮬레이션)<br>석재영 (Ethernet 시뮬레이션 |
| **7월 1-2주** (7.1-7.18) | 보안 알고리즘 설계 및 중간보고서 | - 경량 보안 알고리즘(AES, SPECK) 설계<br>- Simulink에서 초기 알고리즘 통합 테스트<br>- 중간보고서 작성 및 중간평가표 제출 | 홍재왕, 석재영 (알고리즘 설계)<br>레퐁푸 (Simulink 통합, 보고서) |
| **7월 3-4주** (7.19-7.31) | 보안 알고리즘 구현 | - CAN 통신에 메시지 인증(MAC) 구현<br>- Ethernet 통신에 TLS 적용 테스트<br>- 부품별 알고리즘 최적화 시작 | 홍재왕 (CAN 보안)<br>석재영 (Ethernet 보안)<br>레퐁푸 (부품 모델링 지원) |
| **8월 1-4주** (8.1-8.31) | UI 개발 및 알고리즘 비교 | - MATLAB App Designer로 UI 프로토타입 개발<br>- 보안 알고리즘 성능 비교(속도, 리소스, 보안성)<br>- 로그 분석 LLM(LLaMA 3 8B) 파인튜닝 시작 | 레퐁푸 (UI 개발)<br>홍재왕, 석재영 (알고리즘 비교, LLM) |
| **9월 1-2주** (9.1-9.19) | 테스트 및 최종 보고서 작성 | - Simulink에서 공격 시나리오(스푸핑, DoS) 테스트<br>- UI 완성 및 시각화 구현<br>- 최종보고서 및 최종평가표 작성 | 홍재왕, 석재영, 레퐁푸 |
| **9월 3-4주** (9.20-9.30) | 발표 준비 | - 테크위크 발표 자료 준비<br>- 시뮬레이션 및 UI 데모 시연 리허설 | 홍재왕, 석재영, 레퐁푸 |
| **10월 1-2주** (10.1-10.15) | 결과물 업로드 | - 최종 코드, 모델, 보고서 정리<br>- AWS S3에 결과물 업로드<br>- 프로젝트 문서화 완료 | 레퐁푸 (문서화)<br>홍재왕, 석재영 (검토 |

TABLE1. 개발 일정 및 역할 분담

## 2. 역할 분담

구성원별 역할은 프로젝트 목표와 일정에 맞춰 다음과 같이 분담했다.

가. 홍재왕 (CAN 프로토콜 담당): CAN 프로토콜의 메시지 프레임 구조, 데이터 전송 방식, 우선순위 처리 메커니즘을 학습하고, 보안 취약점(메시지 위조, 리플레이 공격)을 분석한다. Simulink에서 CAN 통신 시뮬레이션을 구현하고, 메시지 인증 코드(MAC) 및 경량 암호화(SPECK)를 적용해 보안성을 검증한다. 로그 분석 LLM 파인튜닝과 알고리즘 비교에 참여한다.

나. 석재영 (TCP/IP, Ethernet 담당): Automotive Ethernet과 TCP/IP 프로토콜의 고속 통신 특성과 V2X 통신 활용 방안을 학습한다. TCP/IP 스택의 보안 취약점을 분석하고, Simulink에서 Ethernet 통신 시뮬레이션을 통해 TLS 적용 가능성을 검토한다. 보안 알고리즘 구현과 성능 비교, LLM 기반 로그 분석을 지원한다.

다. 레퐁푸 (Simulink 담당): Simulink 환경 구축과 MathWorks 라이브러리를 활용한 전장부품(ECU, 센서) 모델링을 담당한다. CAN 및 Ethernet 통신 시뮬레이션 환경을 설정하고, 보안 알고리즘 통합 워크플로우를 설계한다. MATLAB App Designer로 UI를 개발하며, 보고서 작성과 결과물 문서화를 주도한다.

위 일정과 역할 분담은 프로젝트의 성공적인 완수를 목표로 하며, 필요 시 유연하게 조정된다.

.

# Ⅷ. 참고문헌

1. TCP/IP & Ethernet

https://westahn.com/osi-4-%EA%B3%84%EC%B8%B5%EC%9D%B4%EB%9E%80/

2. CAN

https://m.blog.naver.com/lagrange0115/221941482740

https://ddongwon.tistory.com/34